The Central Role of the PC in Teaching Statistics at School

Arthur Engel - Frankfurt, Germany

1. Introduction

Usually the PC is used in statistics to do quickly and conveniently what we have always been doing. This is a misuse of the PC since it has the potential to change statistical practice fundamentally. Historically, statistics was developed when computation was hard and expensive. To avoid massive computation a lot of sophisticated theory based on asymptotics was developed. Now computation is cheap and easy. The PC should replace sophisticated theory by simple computations, and make statistics more comprehensible.

We should strive for *understanding*. Sophisticated statistical software is pedagogically harmful. We do not want to solve a dozen problems a day by using recipes. We want to solve a few paradigmatic examples in a few weeks. School statistics should solve a small number of fundamental problems, not quickly, but leisurely. We should derive programs for the solutions, which are general enough, so that they solve a whole class of problems. I do not advocate much deep programming, which is quite difficult. On the other hand, statistical software is for professionals. To learn its use requires an effort comparable to the effort of learning a new programming language. Design of a program is an important part of the learning process. A problem is solved if you have an efficient algorithm for its solution, which you understand.

2. What does statistics do? The two sample problem

In the words of Bradley Efron, statistics compares sets of numbers - with each other, with theoretical models, and with past experience (Efron, 1979). Here is a prototypical statistical question: *Is Method A better than Method B?*

Example #1: I can reach my office via two routes, A or B. By stopwatching the time (in deciminutes) I get, for A: 65, 69, 72, 75, 76, 80, 82, 85, 85, 94, and for B: 74, 79, 79, 81, 86, 88, 95, 97, 99.

Let us subtract the minimum 65 from these numbers. Then we get two sets of numbers A = {0,4,7,10,11,15,17,20,20,29} and B = {9,14,14,16,21,23,30,32,34}.

Is the set A in some sense smaller than the set B? To answer this question we add up the elements in A and B getting $\Sigma A = 133$ and $\Sigma B = 193$. Let us combine the two sets A,B into one set C with 19 elements. Now consider all ${}^{19}C_9 = 92\,378$ partitions of the set C into sets X and Y of 10 and 9 elements, respectively. Find the number, s, of all partitions with $\Sigma X \leq 133$. Then $P = s/92378$ is the probability of getting such an extreme value or even a more extreme one by pure chance if A-roads are not faster than B-roads.

The problem reduces to counting the number q(n,k,t) of k-subsets of the n-set of nonnegative integers $\{d_1,...,d_n\}$ with sum $\leq t$. Obviously

$$q(n,k,t) = q(n-1,k,t) + q(n-1,k-1,t-d_n)$$

with boundary conditions

$$q(n,k,t) = 1 \text{ for } k = 0 \text{ and } t \geq 0 \quad \text{and} \quad q(n,k,t) = 0 \text{ for } n < k \text{ or } t < 0.$$

This recursion can be immediately translated into the Pascal program below:

```
program TwoSample;
const d:array[1..19] of byte=(0,4,7,9,10,11,14,14,15,16,17,20,20,21,23,29,30,32,34);
var n,k,t,s:integer;
function q(n,k,t:integer):integer;
begin
   if (k<0) or (t<0) then q:=0
   else if k=0 then q:=1
   else q:=q(n-1,k,t)+q(n-1,k-1,t-d[n])
end;
begin write ('n,k,t='); readln(n,k,t); s:=q(n,k,t);
   writeln('s=',s,'  P=',s/92378.0)
end.
```

With (n,k,t) = (19,10,133) we get in two seconds s = 2991, P = 3.24%.

Example #2: Should hospitals allow parents of sick children to stay with them to shorten the stay in the clinic (rooming-in model)? To find out, 50 children were selected and partitioned at random into two 25-subsets. For one subset the parents stayed with the children. This subset produced the set A = {6,6,7,7,7,7,8,8,8,8,8,8,8, 10,12,12,14,15,16,17,17,19,21,29,32} of days in the clinic. The other set, B = {7,7,7,7,8,10,11,11,11,12,12,14,15,15,16,18,18,22,23,28,29,29,31,44} of days, was for the children without rooming-in. Let us subtract 6 from each member of the two sets. We get A = {0,0,1,1,1,1,2,2,2,2,2,2,2,4,6,6,8,9,10,11,11,13,15,23,26} with $\Sigma A = 160$ and B = {1,1,1,1,2,4, 5,5,5,6,6,8,9,9,9,10,12,12,16,17,22,23,23,25,38} with $\Sigma B = 270$. Now we combine A and B into one set C and consider all ${}^{50}C_{25} = 12641\,06064\,37752$ partitions of the set C into sets X and Y of 25 elements each. Find the number, s, of all partitions with $\Sigma X \leq 160$. Then $P = s / {}^{50}C_{25}$.

The recursive program becomes unsuitable, and we must translate it into an iterative program, even a sophisticated one, which saves space. The straightforward program which computes the matrix q(y,x,z), requires $\approx n*k*s = 50*25*160 = 200000$ memory locations for real numbers. So much space is usually not available on a PC. The sophisticated approach consists in reducing the number of dimensions to two by considering a fixed y-layer, finding the q(x,z)-values for this layer with the recursion $q(x,z) = q(x,z) + q(x-1,z-d(y))$. If we go downward with x and z we can store the new-found q(x,z)-values in the preceding layer. The resulting program, TwoSamIt below, for $n = 50$, $k = 25$, $t = 160$ takes nine seconds to give $P = 3.4\%$. We need $(k+t)(t+1) + n = 26*161 + 50 = 4236$ real memory locations, which is available on any PC.

```
program TwoSamIt;
const d:array[1..50] of byte=(0,0,1,1,1,1,1,1,1,2,2,2,2,2,2,2,2,4,4,5,5,5,6,6,6,6,8,8,
                              9,9,9,9,10,10,11,11,12,12,13,15,16,17,22,23,23,23,25,26,38);
var min,n,k,t,x,y,z:integer; s:real; q:array[0..25,0..160] of real;
begin write('n,k,t='); readln(n,k,t);
    for z:=0 to t do q[0,z]:=1;
    for x:=1 to k do
    for z:=0 to t do q[x,z]:=0;
    for y:=1 to n do
    begin if k<y then min:=k else min:=y;
        for x:=min downto 1 do
        for z:=t downto d[y] do q[x,z]+q[x-1,z-d[y]]
    end;
    s:=q[k,t];
    writeln('q=',s:0:0,'  P=',s/126410606437752.0*100:0:2,'%')
end.
```

In reality the children were matched in pairs, each pair having the same age, disease, intensity, etc. For each pair it was decided by a coin, which child gets his/her parents into the hospital. The 25 pairs (x_k,y_k) with x_k = number of days in the clinic without R-I and y_k = number of days with R-I for the kth pair were (29,29), (7,7), (12,12), (7,7), (7,7), (10,10), (15,14), (7,6), (18,16), (11,8), (11,8), (11,8), (12,7), (14,19), (22,17), (15,8), (15,8), (29,21), (23,15), (8,17), (18,8), (16,6), (44,32), (28,12), (31,8). For the differences $d_k = x_k - y_k$ we get (0,0,0,0,0,0,1,1,2,3,3,3,5,-5,5, 7,7,8,8,-9,10,10,12,16,23). See Fernandez-Jung (1983) for details. As test statistic we use

$$T = \sum_{k=1}^{25} I_k |d_k|,$$

with the random variable I_k being the kth toss of a good coin with faces 0 and 1.

Null hypothesis H: It is just a random fluctuation. There is no difference between rooming-in children and other children.
Alternative A: Rooming-in helps to shorten stay in the hospital.

If H is true the two negative differences with sum 14 are just due to the coin. So we find $P = P(T \leq 14 \mid H)$. This we generalise to get a problem, which solves all problems with matched pairs. We want to count the number q(n,t) of subsets of the n-set $\{d_1,...,d_n\}$ of nonnegative integers with sum $T \leq t$. We get immediately

$$q(n,t) = q(n-1,t) + q(n-1,t-d_n)$$

with the boundary conditions

$$q(n,t) = 0 \text{ for } t < 0 \quad \text{and} \quad q(0,t) = q(n,0) = 1.$$

The resulting simple and very efficient program Rematch is set out below. Let us pause for a moment to appreciate the progress due to the PC. Formerly the problem of matched pairs was solved by going from the differences to the ranks and applying Wilcoxon's Signed-Rank Test. One looked into a table for small values of n and into the table of the normal distribution for large n. In addition, one had to face the terrible problem of ties and the minor but annoying problem of interpolation. Now we need no tables at all. Each time we compute exactly the P-value. Ties do not play any role. For differences we cannot have tables, because we would have to compute a table for each set of differences, clearly an impossible task. Even more convincing is the program TwoSample, which solved *exactly* a more difficult problem.

```
program rematch;
const d:array[1..19] of byte=(1,1,2,3,3,3,5,5,5,7,7,8,8,9,10,10,12,16,23);
function q(t,n:integer):integer;
begin
    if t<0 then q:=0
    else if (t=0) or (n=0) then q:=1
    else q:=q(t,n-1)+q(t-d[n],n-1)
end;
begin
    writeln('P=',q(14,19)/1024/512)
end.
```

In this program we have dropped the six zero's because they do not matter, but this does speed up the program considerably. In a fraction of a second we get $P = 1.024*10^{-3}$. This is considerably more evidence for the alternative that rooming-in helps to shorten stay in the clinic. This program, or its iterative version, solves exactly any *matched pairs* problem that occurs in practice.

3. The troublesome binomial distribution

Figure 1 shows a spinner. Let p and q = 1-p be the probabilities of the outcomes 1 (success) and 0 (failure), respectively. Any binary word like 100110...01 with x ones and n-x zeros has the same probability p^xq^{n-x}. There are altogether ${}^nC_x$ such words, and so the probability of exactly x successes in n spins (trials) is

(1) $$b(x) = {}^{n}C_{x}p^{x}q^{n-x}, \quad x = 0,1,2,\dots,n.$$

It shows that we are dealing with a function of three variables, n, x, p, and so tabulation is hopeless. In the form (1), b(n,p,x) is difficult to evaluate, even with a PC. So we change (1) into a recursion.

(2) $$b(0) = q^{n}, \quad b(x) = b(x-1)*r*(n-x+1)/x, \quad r = p/q, \quad x = 1,2,\dots,n.$$

With (2) there is another difficulty. We must first evaluate b(0). For n = 126 we get correctly to 11 significant digits $0.5^{n} = 1.1754943508E - 38$, but for n = 127 we get a run time error because 0.5^{127} lies outside the permissible range of Turbo Pascal, which is $2^{-127} < x \le 2^{127}$.

If $x \le 2^{-127}$ we have *underflow*, for $x > 2^{127}$ we have *overflow*. Both are run-time errors because they are discovered during the running of the program. Further, we need an individual probability b(n,p,x) only quite rarely and then only for small n. What we really need is a sum

$$s = b(c) + b(c+1) + \dots + b(d-1) + b(d)$$

To avoid underflow we must take logarithms in (2):

$$\ln b(0) = n*\ln q, \quad \ln b(x) = \ln b(x-1) + \ln r + \ln((n-x+1)/x).$$

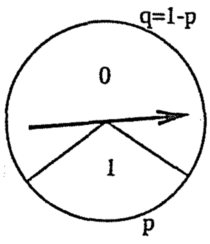

FIGURE 1

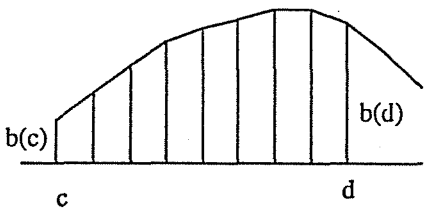

FIGURE 2

Then we can find the sum s in Figure 2 by means of the program below, in which we have set m = n+1. To avoid a run-time error we have introduced in line 8 another precaution. It turns out that $\exp(-88) \approx 2^{-127}$. So for L < -88 we must set exp(L) = 0. Thus we arrive at our final program, BIN, which is very robust.

```
program BIN;
var m,n,c,d,x:integer;
        p,q,r,L,s:real;
begin write('n,p,c,d='); readln(n,p,c,d);
    m:=n+1; q:=1-p; r:=ln(p/q); L:=n*ln(q);
    for x:=1 to c do L:=L+r+ln(m/x-1);
    if L<-88 then s:=0 else s:=exp(L);
    for x:=c+1 to d do
    begin L:=L+ln(m/x-1); if L > -88 then s:=s+exp(L)
    end;
    writeln(s)
end.
```

With this program you can forget the table of the normal distribution as a limit of the binomial distribution. You can find exact P-values instead of dubious approximations. Even more than that: With the program BIN you can find exact confidence intervals for the binomial parameter p, as will be shown in the next paragraph.

A random process is controlled by the spinner in Figure 1 with unknown parameter p. I make n spins and get X successes. Then

$$\hat{p} = X/n$$

is a so-called *point estimate* for p. We show now how one can find bounds such that we can say with a specified level of confidence that p lies within those bounds.

Suppose we get x = 80 successes in n = 200 trials. Then 80/200 = 0.4 is a point estimate for p. Of course we can get 80 or fewer successes even if p is larger than 0.4, but the larger the value of p is the less likely this is. Let us find the value of p for which $P(X \le 80) = 2.5\%$ exactly. The equation for p we must solve is

$$B(p) = P(X \le 80) = \sum_{x=0}^{80} {}^{200}C_x\, p^x\, (1-p)^{200-x} = 0.025.$$

We do this by intelligent search with the program BIN. The inputs n = 200, c = 0, d = 80 and different p-values give $\hat{p}_2 = 0.47147$. The trial steps are given in the table below.

p	0.45	0.47	0.471	0.472	0.4714	0.47147
B(p)	0.088	0.0275	0.02577	0.0241	0.025106	0.0249906

So if $p \ge 0.47147$ the probability of getting only 80 successes in 200 trials is 2.5% or less. We say we can assert with 97.5% confidence that $p < 0.47147$. This means that if we repeatedly make assertions to which a confidence level of 97.5% can be attached in this way, we shall be right, in the long run, a least 97.5% of the time.

By solving the equation (3) below for p we find in a similar way that $0.33155 \le p$ is also a 97.5% interval for p. Therefore the intersection of the two intervals, $0.33155 \le p \le 0.47147$, is a 95% confidence interval. Note that picking equal probabilities for the upper and lower bounds is somewhat arbitrary and it usually does not give the shortest possible confidence interval.

$$C(p) = P(X \ge 80) = \sum_{x=80}^{200} {}^{200}C_x\, p^x\, (1-p)^{200-x} = 0.025. \tag{3}$$

This computer method is the only one which gives exact confidence intervals for p.

4. Hypergeometric distribution. Fisher's Exact Test

For the novel *Quiet Don*, M Sholokhov received in 1965 the Nobel Prize for literature. Since 1928 there were rumours both in the Soviet Union and abroad that the novel is not his work. In an anonymous study by a Soviet critic the novel is attributed to the Cossak writer F Kryukov, who died of typhoid in 1920. (See Kjetsaa for details.)

1000 words were chosen at random from each of *Marking Time* (Kryukov), *The Way and the Road* (Sholokhov) and *Quiet Don* (?). Each time the number of lexemes (different words) was counted. A *lexeme* is a dictionary entry. For instance: *write*, *writes*, *wrote*, *written*, *writing* are the same lexeme. The following table shows the result.

Text	Words	Lexemes (different words)
Marking Time (Kryukov)	1000	589
The Way and the Road (Sholokhov)	1000	656
The Quiet Don (?)	1000	646

Since there is almost no difference between Sholokhov and the author of the *Quiet Don*, we test Kryukov versus the author of the *Quiet Don*, based on the 2×2 table below.

Text	1000-L	Lexemes L	Sum
Quiet Don	354	646	1000
Kryukov	411	589	1000
Sum	765	1235	2000

H: Kryukov could have written the *Quiet Don*.
A: Kryukov did not write the *Quiet Don*.

If H is true the probability of getting any particular table

x	1000-x	1000
765-x	235-x	1000
765	1235	2000

is given by $h(x) = {}^{1000}C_x\ {}^{1000}C_{765-x}/{}^{2000}C_{765}$. The probability of such an extreme or even more extreme result is

$$P = \sum_{x \le 354} h(x) = 0.00498 \approx 0.005.$$

This P-value was computed with the program *Left_Tail*, which calculates the probability h(x).

$$h(x) = {}^{r}C_{x}\,{}^{n-r}C_{s-x} / {}^{n}C_{s}$$

using

$$h(0) = {}^{n-r}C_{s} / {}^{n}C_{s} = \frac{(n-r) \ldots (n-r-s+1)}{n \ldots (n-s+1)},$$

and the recursion

$$h(x) = h(x-1)\,\frac{(r-x+1)(s-x-1)}{x(n-s-r+x)}.$$

```
program left_tail;
var i,n,r,s,x:integer; L,p:real;
begin write('n,r,s,x='); readln(n,r,s,x); L:=0; p:=0;
    for i:=1 to s do L:=L+ln((n-r-i+1)/(n-i+1));
    if L>=-88 then p:=exp(L);
    for i:=1 to x do
    begin
        L:=L+ln((r-i+1)/i*(s-i+1)/(n-r-s+i));
        if L>=-88 then p:=p+exp(L)
    end;
    writeln('P=', P)
end.
```

For the *Quiet Don* the input was n = 2000, r = 1000, s = 765, x = 354. Running time was 4 seconds on a 12 MHZ AT. Note that we have used logarithms to avoid underflow, as in program BIN.

5. The permutation test and correlation

I have shown (Engel, 1985) how to treat Kendall's τ in a computer oriented way. Let us take the same example which I will treat differently. The vector X = (1.25, 1.62, 2.49, 2.57, 3.41, 3.83, 6.41, 8.34, 11.64) gives, for nine Oregon counties, the index of exposure to radioactivity and Y = (113.5, 137.5, 147.5, 130.1, 129.9, 162.3, 177.9, 210.3, 207.5) is the corresponding cancer mortality (per 100000 man years for 1959-1964). The components of X are sorted increasingly. This seems to cause the Y-components also to increase in general.

We want to find out if this could be due to chance. With the computer we can take a test statistic of our own invention. For instance, let us take the dot product T of X and Y, i.e.

$$T = XY = \Sigma X[i]Y[i].$$

Note that the components of X are sorted increasingly. T is a maximum if also the components of Y are sorted increasingly. Because of some inversions the observed value Cor of T is smaller than the maximum. We first find Cor = 7440.3660. Now we permute the components of Y *at random* and call the permuted vector Z. This is repeated 10000 times, and each time dot(X,Z) ≥ Cor we set count ← count+1. Since it is difficult to estimate very small probabilities, we have repeated the program ten times getting the count-values 5, 2, 3, 4, 2, 3, 5, 2, 3, 5 and the estimate $\hat{P} = 0.00034$.

```
program cancer1; const n=9; type vector=array[1..n] of real;
const   X:vector=(1.25, 1.62, 2.49, 2.57, 3.41, 3.83, 6.41, 8.34, 11.64);
        Y:vector=(113.5, 137.5, 147.5, 130.1, 129.9, 162.3, 177.9, 210.3, 207.5);
var count,i,j,r:integer; Cor,copy:real; Z:vector;

function dot(S,T:vector):real;
var i:integer; sum:real;
begin sum:=0.0;
    for i:=1 to n do sum:=sum+S[i]*T[i];
    dot:=sum
end;

begin Cor:=dot(X,Y); writeln(Cor:0:4); count:=0;
    for j:=1 to 10000 do
    begin Z:=Y;
        for i:=n downto 2 do
        begin r:=1+random(i); copy:=Z[i];Z[i]:=Z[r];Z[r]:=copy end;
        if dot(X,Z)>=Cor then count:=count+1
    end;
    writeln(count)
end.
```

If we use *Pearson's Correlation Coefficient* as test statistic, the program will be much more complicated. Ten repetitions of the corresponding program yielded the count numbers 5, 2, 5, 6, 3, 5, 3, 0, 4, 2. The observed significance level 0.00035 is practically the same. It would be possible to step through all 9! = 362880 permutations of the components of Y at the expense of a more complicated program. Instead, we have settled for a random subset of these permutations. For large n complete enumeration is not feasible anyway.

6. Conclusions

In summary we can state that statistical tables are no longer needed. The teacher may cling to the excuse: "But the PC is not available to everyone!". Most of this can be done with a programmable pocket calculator, which costs no more than an expensive book. Anyway, in 2 or 5 or 10 years the pocket calculator will have the power of a PC. The only thing that stops a PC are the *intractable* problems. For these simulation gives results. Most of the permutation tests are *NP-hard*. This is only a small disadvantage.

After all, statistical numbers are small, and for small numbers the algorithms are *pseudo-polynomial*, as was shown for the programs *TwoSample* and *Rematch*, two famous NP-hard problems.

References

More details on the computer oriented approach to statistics can be found in Engel (1990) and Engel (in press). Engel (1990) has just been published. About one-third of it is devoted to probabilistic and statistical problems. The remainder is devoted to problems in pure mathematics. Engel (in press) is an improved English version of Engel (1990). It will appear in 1991. All examples in my paper are from the "in press" version. Most of them you can also find in the 1990 version. When I decided to revise my textbook (Engel, 1972) I had to relearn some classical statistics. The result was Engel (1987). I missed an opportunity to write a textbook on statistics with a PC. But I thought the time was not yet ripe for such a venture.

Efron, B (1979) Computers and the theory of statistics : thinking the unthinkable. *SIAM Review* **21(4)**.

Engel, A (1972) *Wahrscheinlichkeitsrechnung und Statistik, Band 1.* Ernst Klett Verlag, Stuttgart.

Engel, A (1985) Statistics and computer science : an integrated high school course. In: L Rade and T Speed (eds) *Teaching Statistics in the Computer Age.* Chartwell-Bratt Ltd.

Engel, A (1987) *Stochastik.* Ernst Klett Verlag, Stuttgart.

Engel, A (1990) *Mathematisches Experimentieren und Statistisches Simulieren mit Einem PC.* Ernst Klett Verlag, Stuttgart.

Engel, A (in press) *Mathematical Experimentation and Statistical Simulation on a PC.* New Mathematical Library, Mathematical Association of America.

Fernandez-Jung, F (1983) *Auswirkung elterlicher Mitaufnahme (rooming-in model) auf das Verhalten stationärehandelter Kinder.* Diss. FU Berlin.

Hayman, R (1985) The Ganzfeld experiment : a critical appraisal. *Journal of Parapsychology* **49**, March, 3-49.

Kjetsaa, G The battle of the Quiet Don : another pilot study. *Computers and Humanities* **11**, 341-346.